Proceedings of the 2018 IEEE/ASME International
Conference on Advanced Intelligent Mechatronics (AIM),
Auckland, New Zealand, July 9-12, 2018

Interactive Calibration and Visual Programming of Reconfigurable Robotic Workcells

Marc Priggemeyer, Daniel Losch and Jürgen Roßmann
Institute for Man-Machine Interaction
RWTH Aachen University
Ahornstraße 55, 52074 Aachen, Germany

{priggemeyer, losch, rossmann}@mmi.rwth-aachen.de

***Abstract*— Small and Medium Enterprises can utilize robotic assembly solutions to improve their productivity. This is especially true if the assembly cells can be reconfigured to accommodate smaller batch sizes and flexible tasks. The Horizon 2020 project "ReconCell" aims to develop such a reconfigurable work cell. We created a Digital Twin of this reconfigurable robot cell in order to follow the Industry 4.0 approach. Users can visually model and develop different assembly processes for the digital reconfigurable robot cell, as well as execute them in simulation. To be able to transfer the simulated processes to hardware, the Digital Twin has to be consistent with the setup of the real robot cell: It needs to be thoroughly calibrated before it can be used. In this contribution, we present methods to conduct such a calibration interactively, and use the resulting consistency to conduct virtual commissioning of simulated assembly processes and transfer such processes to a physical robot cell.**

I. Introduction

Reconfigurable workcells allow fast change-over times and easy changes in automated manufacturing processes. Especially Small and Medium Enterprises (SME) are targeted for this kind of workcell since these companies usually have a portfolio of highly specialized parts and components with a very low level of automation in their manufacturing processes [1]. In many cases, costs for experts to build a workcell that meets an SMEs requirements are too high. This holds true not only for initial costs, but also for recurring costs if changes in the process are necessary. A more viable solution is to include SMEs in the process of setting up robotic systems and allow them to make changes to the systems on their own. The inclusion of the company can happen in three different phases:

1) The SME can create an initial workcell layout in a virtual environment and do the calibration on the real workcell.
2) The SME can do the actual programming of tasks and thus create and adapt the process it wants to automate.
3) The SME can do recurring tasks like changeover and recalibration.

Due to the fact that it is desirable to have non-expert users on site that are capable of performing these changes, ease of use with regard to the hardware and software interfaces for reconfiguration tasks is a necessary requirement. That implies the utilization of tool exchange systems and versatile actuators that can be utilized in a variety of processes. In addition, structural parts and machine elements that can be easily modified and assembled in different layouts are desirable. This conforms to the idea of easily operated Plug-and-Produce components [2] that are mandatory for successful deployment and commissioning of flexible robotic systems.

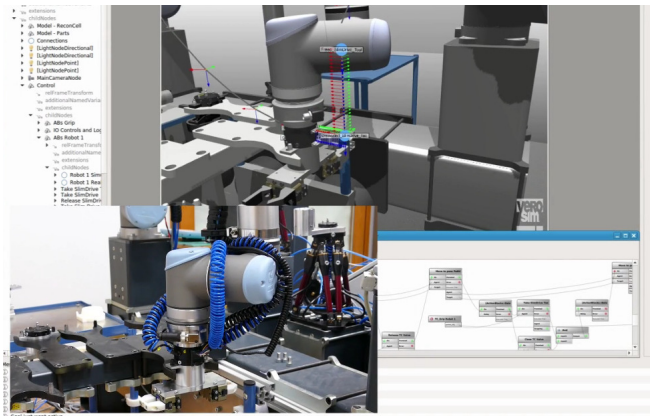


Fig. 1: Visually Programmed tool exchange process in a virtual and physical environment.

With reference to the hardware interfaces, it is important to distinguish two different kinds of reconfiguration tasks that allow for workcell design changes: The first task includes all changes that demand a shutdown while workers can manually dis- and reassemble a workcell, while the second one includes automated changes that are implemented by (e.g.) passive fixtures or actuators [3]. Gaspar et al. presented an exemplary system (ReconCell) comprising different types of reconfigurable hardware and a software infrastructure to do low-level operations like actuator movements and a number of automated reconfiguration tasks in the workcell [4].

Considering the programming capabilities of conventional robot controllers, a new approach is necessary for reconfiguration tasks and easy adaptation of processes to create complex, while highly flexible programs in manufacturing applications. From a conventional controllers perspective, such a reconfiguration task is quite complex. Such a con-

978-1-5386-1854-7/18/$31.00 ©2018 IEEE

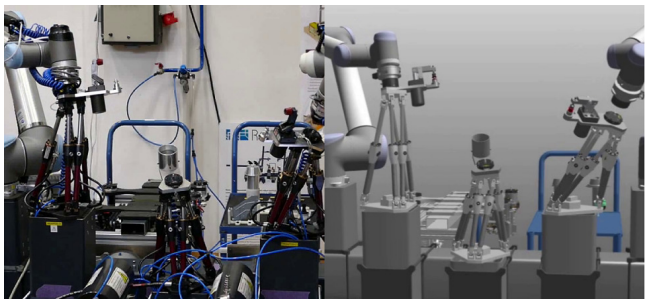


Fig. 2: Side-by-side view of a real ReconCell (left) and its digital twin (right).

troller demands a large number of poses that have to be taught manually. This also implies that the controller needs to manage a large number of configuration states of a workcell. Therefore, we require a software environment that includes all workcell elements and allows for process execution control while maintaining a minimal set of reference points to deduce workpiece and element poses in a workcell. Virtual Testbeds (VTB) fulfill the requirements by offering a virtual environment that can be used for initial planning, programming, testing and final execution [5]. A VTB is executed by a simulation system that provides a number of advantages considering different domains like (Rigid Body Dynamics, Sensor Simulation, etc.) that can be utilized for a holistic system model. Thus, a workcell can be created in the VTB before it is deployed to the shop floor. This includes the design of an initial workcell layout integrating structural elements, flexible fixtures, actuators and workpieces. The entirety of all these parts in a structured form is called simulation model that can be loaded into the simulation system and thereby integrated into the virtual environment. Virtual Testbeds are also a foundation for Simulation-based Control that allows to transfer simulation results from the virtual environment to the physical workcell. Figure 2 shows a view on an exemplary workcell in the real and in the virtual environment. The digital twin of the real workcell is employed either independently to verify e.g. motion planning, or it is closely coupled to transfer motion commands to the actuators while mirroring the cell behavior. Since all the workcell elements are integrated in a central control instance, the virtual environment can be utilized to do optimizations of (e.g.) workpiece placements as shown by Atorf [6].

In the following sections, we will summarize how the ReconCell infrastructure can be utilized for high-level planning tasks. We will briefly outline how a manufacturing process can be defined by a programming paradigm that allows for an easy transfer from a workcell in the virtual environment to the physical cell. Since this requires a consistent representation of the real cell in the virtual environment, we will further specify how the calibration process of cell elements works.

II. A Reconfigurable Workcell

We apply the presented techniques to a reconfigurable robotic workcell, which is called ReconCell. An in-depth description of all the hardware and backend software components can be found in [4]. A ReconCell consists of a steel frame, a set of passive fixtures and a number of actuators, i.e. two industrial robots, a tool exchange system, grippers and other tools, either specialized to a specific use case or for general purpose usage. The hardware is controlled by a software backend that allows an operator to issue movement commands and trigger binary actions like gripping or activating a screwdriver. The ReconCell infrastructure defines a set of messages in a ROS network that provides detailed information on the current cell status. In addition, distinct action servers are provided to offer a set of services that can be triggered by the cell operator. Some examples of such services include Point-to-Point and Linear Movements, or actions to trigger sensor data processing components to identify the quality of operations that were carried out on a workpiece [7].

The virtual environment acts as a frontend for a ReconCell to enable an operator to execute high-level tasks. The integration of virtual and real environment is realized in two parts.

A. State replication in Virtual Testbeds

The ReconCell VTB is capable of accumulating all the status messages in the ReconCell ROS network to replicate executed tasks in real-time. Spatial information of the robots' TCPs is used to identify grippers and tools that need to be triggered during gripping actions. Each robot's position is updated continuously and mirrors the real robot. Due to the fact that the real cell's setup mirrors the simulation, the spatial information of all parts is available to identify the part an action is to be carried out on. Gripping actions are identified by the logical state change of the robot I/O and are also mirrored in the VTB by triggering the gripper's digital twins.

Thus, the analysis of cell operation becomes possible by investigating the replicated states in the virtual environment. This opens up a wide range of possibilities for testing, optimization and verification, as all robot states can be displayed during any task in the workcell. Using this method, we can easily collect operational data for further analysis.

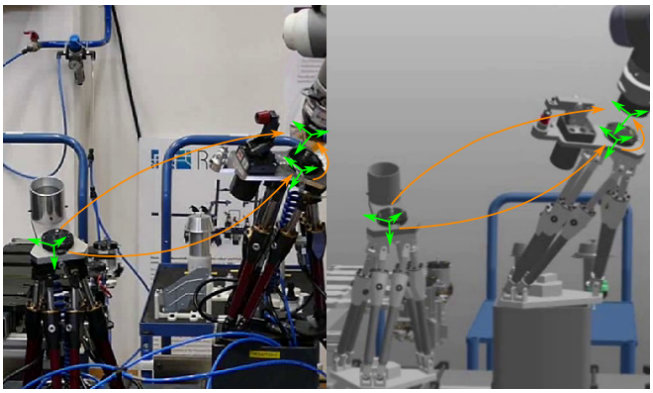


Fig. 3: Robotic workcell with frames used for calibration.

B. Simulation-based Control

In addition to reviewing the behavior of the digital twin during simulated cell operations, we can instruct services to control the cell's real actuators and review the behavior of the physical twin. In order to connect the real twin to our simulation system, we embedded clients that correspond to Action Servers in the ROS network into the VTB. The ActionBlock concept, which is further outlined in the next section, introduces agents that utilize this embedded clients in the ReconCell backend. Simulation-based Control is essential to operate the cell to its full capabilities. Based on the simulated virtual environment, a cell design is used to do the planning and execution of robot motion. The overall benefit lies in the monitoring of the comprehensive cell state and the generated information that can be used for automatisms not available in conventional work cells.

Both cases require calibration of the digital twin to the state of the real cell. We explain the calibration process in detail in Section IV. However, Figure 3 contains a preliminary example of such a calibration: On both sides of the picture, three frames are marked. Two are located on the passive fixtures' (i.e. hexapods') top plates and one is located at the tool center points (TCP) of the robot to the cells' right. We can determine transformations between each pair of the three frames in the real cell and each pair of the three frames in the virtual cell. Those transformations are here portrayed by arrows. The cell is calibrated when the transformations in the real and virtual cell match.

As soon as the virtual robot's TCP touches one of the hexapods top plates, it is essential that the real robot's TCP does likewise. Otherwise, fatal collisions might occur that cause damages to the hardware and occasion high costs for repair and downtime of the workcell. This applies to the case that the workcell is controlled from the virtual environment as well as the case that operations are passively reconstructed in the virtual environment.

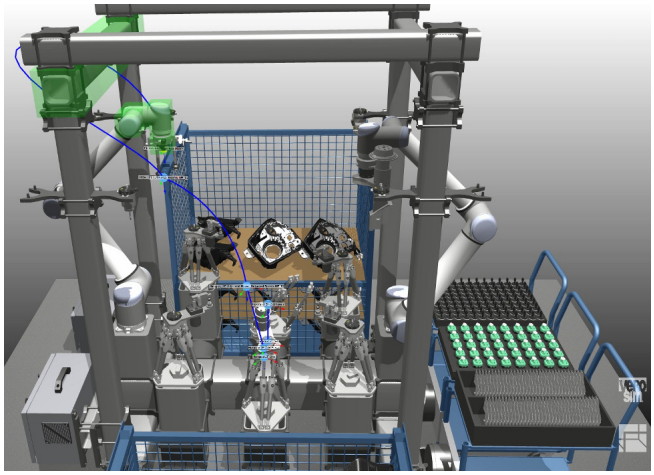


Fig. 4: Analysis of robot motion during cell setup in the virtual environment revealing collisions between actuator and frame.

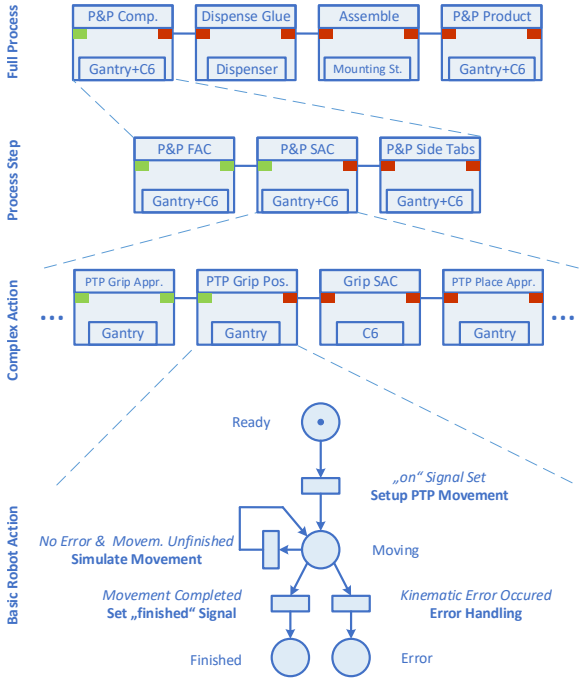


Fig. 5: Exemplary process, visually modelled and programmed by a hierarchy of ActionBlocks (taken from [8]).

In ReconCell, a calibrated workcell is also integrated into the development workflow, such that the operator gets steady visual feedback while planning actuator motion during programming tasks. Figure 4 depicts an automatically planned path that would lead to a collision between the highlighted (green) cell elements. In a calibrated state, the operator can ensure that a collision free path found in the VTB will not generate a collision in the real workcell.

III. Visual Programming

In contrast to conventional, text-based programming methods, Visual Programming is a graphics-based paradigm based on visualizing and manipulating semantic connections between items. Visual Programming is usually utilized by inexperienced developers or non-experts, and offers its users a concrete, direct, explicit way of programming while they experience immediate visual feedback [9]. Examples of a Visual Programming environments for robot control are LEGO Mindstroms [10] and RoboStudio [11].

Schlette et al. developed an agent-based Visual Programming environment for eRobotics (called *ActionBlocks*) and utilized it for micro-optical assembly process development [12] and virtual commissioning [13]. The ActionBlock Visual Programming system is currently used in ReconCell[14]. Losch and Roßmann argued that ActionBlock networks can be mapped to the widespread process modelling methodology of Petri nets [8]. They also presented a formalism to utilize macro-based, recursive ActionBlock compositions as a tool for hierarchical process modelling, where different abstraction levels are mapped to sub-networks of ActionBlocks

(as shown in Figure 5).

In order to use ActionBlocks not only for the design of simulated robot processes, we extended the capabilities of robotic agents and enabled them to interface with the ReconCell ROS network described in section II. The interface is built by ROS Action Clients. Figure 6 displays the AgentSwitcher, a tool to choose between simulated and real execution of an ActionBlock based robot process. From the view of the ActionBlock network, the simulated and real behavioural implementations are functionally equivalent and offer an identical interface. When set to real execution, the robotic agent sends preconfigured ROS command messages to the robot control server and receives its status messages for process monitoring and error handling. This mechanism fully implements the eRobotcs concept of Simulation-based Control. We prototypically implemented ROS-based behaviours for most common robotic actions - movement, gripping and releasing of objects. With just these behaviours, we can already visually program pick-and-place subprocesses, and execute them on a real robot after validating them in simulation. Such an example is depicted in Figure 1:

The top right part shows the simulated ReconCell including a visually programmed path that the robot's TCP traverses. The lower right part shows an ActionBlock program sequence that controls the cells behaviour by combining the movement, grasping and releasing actions. All the agents are defined in the simulation database visible on the top left of Figure 1. They specify whether the actions are to be carried out purely in the VTB or transferred to the real ReconCell. The result is shown in the lower left part that includes the real ReconCell operating synchronously controlled by its digital twin.

Besides the pure logical definition of the control flow represented by ActionBlocks, the graphical cell design and interactive programming is an important step to an easy to use tool kit including interaction capabilities with the virtual environment.

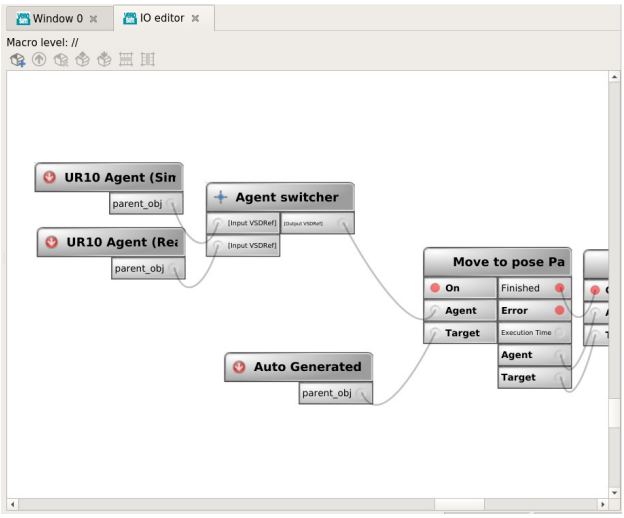


Fig. 6: Visual Programming component to switch between simulated and physical execution of ActionBlocks.

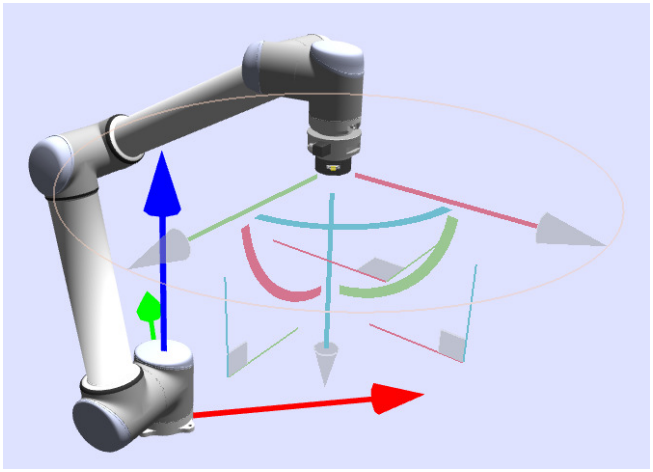


Fig. 7: Kinesthetic guidance feature in the Virtual Testbed

The goal is the simplification of operating the Virtual Testbed during programming tasks (i.e. ease-of-use). Since kinaesthetic guidance is a widely adopted method of teaching poses by demonstration [15], the same behaviour was implemented for the Virtual Testbed. For example, it is beneficial to guide the robot to a tool or workpiece while immediately getting feedback on the feasibility of the action (i.e. reachability, potential collisions).

In Figure 7 an industrial robot is show that can be manipulated by an operator in a constrained way. Two coordinate systems are displayed, one is the base frame that can be used to place the robot at a desirable pose. The second is located at the robot's tool center point offering a set of handles that can be used like classical drag and drop actions. While holding one of the handles, the TCP can be moved along the selected degree of freedom.

IV. Workcell Calibration

In order to utilize Simulation-based Control, we require a sufficiently detailed digital twin of the ReconCell, as seen in Figure 2, and a set of calibration data. We distinguish between two model variants of a ReconCell in the virtual environment, the base model and the calibrated model:

A base model represents the cell during all of the planning and optimization phases. It includes all elements and depicts the ideal workcell that is to be built and commissioned (Figure 8, top left). One of the problems that arise from this is that cell assembly is not an ideal process, and therefore, offsets and inaccuracies between elements (e.g. cell frame beams) can occur.

In contrast, a calibrated model represents the cell state in the virtual environment after it is commissioned (Figure 8, top right). The base model is adapted and all the element positions are corrected to comply with the physical cell (Figure 8, lower left). These corrections are included in a calibration dataset (Figure 8, center) and originate from the difference between the base model and real cell.

For calibration, we identify and associate poses of individual parts both in the physical and in the virtual workcell. We identify two sets of frames $\{{}^{P}F_i\}$ and $\{{}^{V}F_j\}$, with superscript P for the physical cell and superscript V for the virtual cell. Association between frames is appointed by the indices, such that a virtual frame ${}^{V}F_k$ correlates to a physical frame ${}^{P}F_k$.

Between any two poses m and n in the real workcell, we calculate a homogeneous transformation ${}^{P}_{m}T_n$. The transformation must match the transformation ${}^{V}_{m}T_n$ between the corresponding poses m and n in the virtual workcell up to a high degree of accuracy:

$$ {}^{P}_{m}T_n \cdot {}^{V}_{m}T_n^{-1} \rightarrow \boldsymbol{I} $$

For a quantitative analysis, one can either evaluate a metric on the Special Euclidean group $SE(3)$ or, after decomposing both transformations into translations $\boldsymbol{p}$ and rotations $(\boldsymbol{v}, \alpha)$, the Euclidean norm [16].

The previously explained connection holds true for the calibrated model, but it does not clarify how to generate calibration data and augment the base model corrspondingly. To realize the calibration of a base model, we create pose measurements in the real workcell with reference to an arbitrarily chosen base frame. As a result, we determine a transformation for any measurement in a frame ${}^{W}F$. This calculation requires knowledge of the forward transformation between ${}^{W}F$ and any conducted measurement. There are different methods to determine these forward transformations:

Ostrowska et al. showed the feasibility of applying standard 6-axis industrial robots as measurement systems [17]. Since at least one industrial robot is deployed in a ReconCell, we utilize it as the measurement system for the calibration process. The benefit lies in the straight forward description of the serial kinematic chain with Denavit-Hartenberg parameters [18]. These are used in the simulation system to model the actuator to accurately map its mechanical structure during

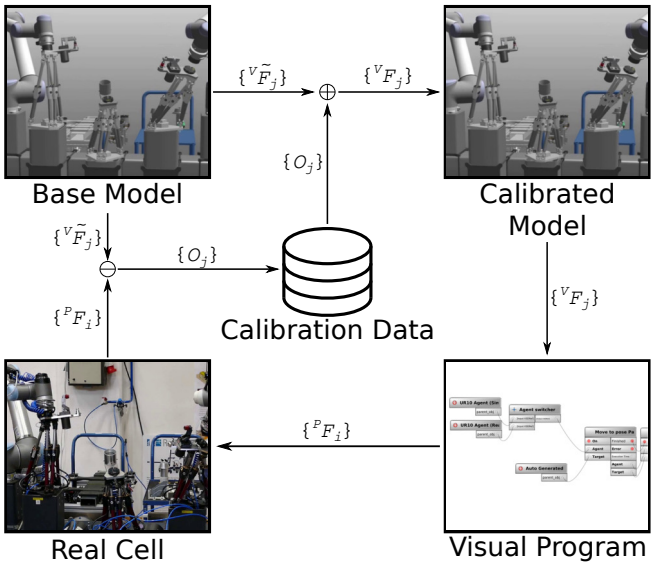
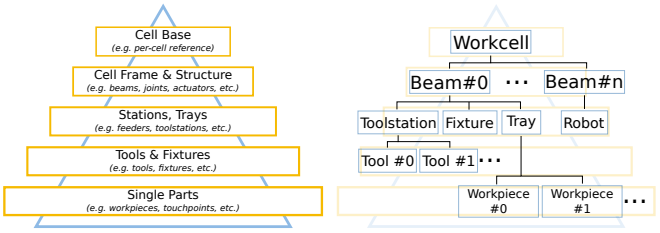


Fig. 8: Creation and application of calibration data.



(a) Hierarchical Calibration Scheme (b) Example Cell Structure

Fig. 9: Scheme and example of hierarchical workcell calibration in the virtual environment.

robot motions.

Due to the fact that we measure axis positions during cell state replication (see Section II-A) we deduce the pose of the tool center point ${}^{M,n}F$ with reference to the robot's base, which we choose to be ${}^{W}F$. Thus, we get the transformation ${}^{P}_{W}T_{M,n}$ from the base to the nth frame in the set $\{{}^{P}F_i\}$. Likewise, ${}^{V}_{W}T_{M,n}$ represents the transformation from the base to the nth frame in the set $\{{}^{V}F_j\}$, since we can also calculate the pose of the tool center point in the virtual environment. The set $\{{}^{V}\tilde{F}_j\}$ includes the frames of the base model that are uncalibrated and analogously we define ${}^{V}_{W}T_{\tilde{n}}$.

Given these transformations, a calibration data element is therefore equal to the offset

$$ O_n = {}^{V}_{W}T_{\tilde{n}}^{-1} \cdot {}^{V}_{W}T_{M,n} $$

Calibration, as described above, is carried out on a per workcell basis and includes the exact spatial information for all the components in that one physical instance of a cell. If, for example, a company operates multiple workcells for the same task in a production facility in order to parallelize production, each workcell gets its own calibration dataset that is applied to the common base model. In this scenario, it is important to reduce the number of frames required for calibration, since maintenance costs for recalibration grow linearly with this number.

Therefore, we conduct a hierarchical, constrained calibration of the (alrady hierarchically structured) base model as shown in Figure 9. Changes on one subtree, i.e. modification of the cell configuration, necessitates only the recalibration of that subtree's root node. Therefore, we reduce the effort of maintenance after manual reconfiguration tasks. Figure 10 shows the simulated environment of a ReconCell overlayed with the current calibration state of the workcell. It helps identifying which parts need to be calibrated (red), which have already been calibrated (green) and which parts are currently in the progress of calibration.

Hierarchical calibration is a handy tool in conjunction with Visual Programming, as it provides graphical feedback on which parts can reliably be used when issuing motion commands. While visually programming a process, the spatial position of tools and parts might only approximated (i.e. uncalibrated). In this case, programming a process is possible, it is essential to verify the resulting movement commands after calibration to avoid collisions and ensure safety. Since

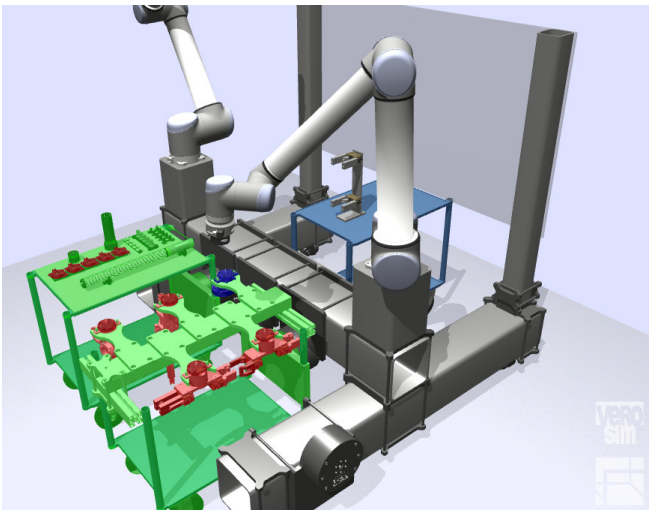


Fig. 10: Creation and application of calibration data. Calibrated components are tinted green, while uncalibrated components are tinted red.

measured offsets are applied to the base model, including an inherent update of target poses for robot movements, transitioning between calibrated and uncalibrated state is seamless. Thus, operators can easily deploy the ReconCell and corresponding programmed processes.

V. Conclusion

In this paper we demonstrated how Visual Programming and interactive calibration can be used to simplify and enhance the user experience during reconfiguration tasks in robotic workcells. With these techniques, Small and Medium Enterprises can improve their productivity due to lower costs during initial setup and maintenance tasks in reconfigurable workcells. This holds especially for small-batch production of highly specialized products. We presented a method for interactive calibration of individual cell elements that supports the transition of robot programs from a virtual environment to the shop floor. We tackled the problem of misalignment of structural elements and tolerances during setup by utilizing calibration, thus avoiding inconsistencies between finals pose of workpieces and fixtures between simulation and reality. The tools and techniques that come with the Visual Programming paradigm further simplify the work of operators and enables process development by non-expert users.

We used the ReconCell as a demonstrator for the presented techniques and thus verified how the goal of an easily programmable robotic workcell can be achieved by introducing a Virtual Testbed that incorporates a Digital Twin of the workcell. The continuous information on the cell state during operation helps to identify issues early and avoid cost-intensive downtimes and repairs.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No H2020-FoF-680431 (ReconCell).

References

[1] Teknologisk Institut. (2015) Robotter i global kamp. https://www.teknologisk.dk/_/media/61776_Industrirobotter%20-%20analyse_teknologisk_instiut.pdf (visited on 2018-02-18).

[2] D. Gorecky, S. Weyer, A. Hennecke, and D. Zühlke, "Design and instantiation of a modular system architecture for smart factories," *IFAC-PapersOnLine*, vol. 49, no. 31, pp. 79–84, 2016.

[3] M. Bem, M. Deniša, T. Gašpar, J. Jereb, R. Bevec, I. Kovač, and A. Ude, "Reconfigurable fixture evaluation for use in automotive light assembly," in *Advanced Robotics (ICAR), 2017 18th International Conference on*. IEEE, 2017, pp. 61–67.

[4] T. Gaspar, B. Ridge, R. Bevec, M. Bem, I. Kovač, A. Ude, and Ž. Gosar, "Rapid hardware and software reconfiguration in a robotic workcell," in *Advanced Robotics (ICAR), 2017 18th International Conference on*. IEEE, 2017, pp. 229–236.

[5] J. Rossmann and M. Schluse, "Virtual robotic testbeds: A foundation for e-robotics in space, in industry-and in the woods," in *Developments in E-systems Engineering (DeSE), 2011*. IEEE, 2011, pp. 496–501.

[6] L. Atorf, C. Schorn, J. Rossmann, and C. Schlette, "A framework for simulation-based optimization demonstrated on reconfigurable robot workcells," in *Systems Engineering Symposium (ISSE), 2017 IEEE International*. IEEE, 2017, pp. 1–6.

[7] T. Ivanovska, S. Reich, R. Bevec, Z. Gosar, M. Tamousinaite, A. Ude, and F. Wörgötter, "Visual inspection and error detection in a reconfigurable robot workcell: An automotive light assembly example."

[8] D. Losch and J. Roßmann, "Visual Programming And Development Of Manufacturing Processes Based On Hierarchical Petri Nets," in *2016 3rd Intl. Conference on Soft Computing & Machine Intelligence (ISCMI 2016)*. Dubai, UAE: IEEE, 2016, pp. 154–158.

[9] M. Burnett, "Software engineering for visual programming languages," *Handbook of Software Engineering and Knowledge Engineering*, vol. 2, 2001.

[10] S. H. Kim and J. W. Jeon, "Programming LEGO Mindstorms NXT with visual programming," in *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*. IEEE, 2007, pp. 2468–2472.

[11] C. Datta, C. Jayawardena, I. H. Kuo, and B. A. MacDonald, "RoboStudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2352–2357.

[12] C. Schlette, D. Losch, and J. Roßmann, "A Visual Programming Framework for Complex Robotic Systems in Micro-Optical Assembly," in *ISR/Robotik 2014; Proceedings of the 41st International Symposium on Robotics;*. VDE, 2014, pp. 1–6.

[13] C. Schlette, D. Losch, S. Haag, D. Zontar, J. Roßmann, and C. Brecher, "Virtual commissioning of automated micro-optical assembly," in *SPIE LASE*. International Society for Optics and Photonics, 2015, pp. 93 460I—-93 460I.

[14] C. Schlette, E. G. Kaigom, D. Losch, G. Grinshpun, M. Emde, R. Waspe, N. Wantia, and J. Roßmann, "3D simulation-based user interfaces for a highly-reconfigurable industrial assembly cell," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–6.

[15] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[16] C. Belta and V. Kumar, "Euclidean metrics for motion generation on se (3)," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 216, no. 1, pp. 47–60, 2002.

[17] K. Ostrowska, R. Kupiec, M. Kowalczyk, P. Wojakowski, H. Skorska, and J. Sladek, "Application of industrial robot as a measuring system," in *Advances in Manufacturing*. Springer, 2018, pp. 797–804.

[18] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.